Conway's cosmological theorem and automata theory

Pierre Lairez, Aleksandr Storozhenko

Conway's cosmological theorem and automata theory

PIERRE LAIREZ, Inria, Université Paris Saclay, France
ALEKSANDR STOROZHENKO, École polytechnique, France

John Conway proved that every audioactive sequence (a.k.a. *look-and-say*) decays into a compound of 94 elements, a statement he termed the *cosmological theorem*. The underlying audioactive process can be modeled by a finite-state machine, mapping one sequence of integers to another. Leveraging automata theory, we propose a new proof of Conway's theorem based on a few simple machines, using a computer to compose and minimize them.

1 INTRODUCTION

In 1986, John Conway published his study of integer decay under *audioactive* derivation [2, 3], veiled in a brilliant atomic metaphor. The derivation process, now well-known in recreational mathematics, mimics how we read strings. For instance, the seed "55555", which we read as "five fives", is derived to "55". In turn, "55" yields "25", iteratively generating the audioactive sequence:

$$55555 \to 55 \to 25 \to 1215 \to 11121115 \to 31123115 \to \dots$$

Any string of numbers, or *word*, such as "55555", may be the start of an audioactive sequence[1].

We say that a word *splits* as the concatenation uv of two contiguous subwords u and v if, for all $k \geq 0$, the kth audioactive derivation of uv is equal to the concatenation of the kth derivations of the subwords. A nonempty word that does not split is called an *atom*, and it is clear that every nonempty word either splits into atoms or is an atom itself. There exist infinitely many distinct atoms. While most only emerge through the derivation of carefully selected seeds, Conway identified exactly 92 atoms, termed the *common elements*, that appear in the derivation sequence of every word except "22" and the empty word. He further identified two families of atoms, the *transuranic elements*, which appear in the derivation of all words containing a digit $d \geq 4$.

Celebrated by Conway as the finest achievement of "audioactive chemistry", the *cosmological theorem* states that there exists some $N \geq 0$, such that for every word x and all $k \geq N$, the kth audioactive derivation of x splits into common and transuranic elements. An interesting corollary is the *arithmetical theorem*: the length of the kth derivation of any given nonempty word, other than "22", exhibits geometric growth with ratio $\lambda \simeq 1.303557$, an explicit algebraic number (see [2] for more details). The original proofs of the cosmological theorem, by Conway, Richard Parker, and Mike Guy, claiming a bound $N = 24$, has been lost, but complete proofs have since been given [4, 6, 10].

The audioactive derivation is (almost) described by a type of finite-state machine, called a *transducer*. It is beyond question that Conway, and all others who have subsequently studied audioactive decay, knew about automata theory and the associated formulation of the derivation process. Yet, none of the published proofs make use of it. We propose to fill this gap, leading to a very simple proof of the cosmological theorem based on two cornerstone results of automata theory: first, the composition of two transducers is a transducer; second, there exists an algorithm to check whether two automata recognize the same language (see Section 2).

Our proof strategy closely follows that of Conway, but with a substantial modernization: whereas Conway's method relied on manually "tracking a few hundred cases" [2, p. 14], we harness the expressive power of automata and transducers, delegating the intricate casework to standard

[1]The strings appearing here should be understood as a sequence of numbers, which most of the time are digits. For example, we note that the audioactive derivation of "2222222222" should really be the two-element sequence $(10, 2)$, not $(1, 0, 2)$.

Authors' addresses: Pierre Lairez, Inria, Université Paris Saclay, Palaiseau 91120, France, pierre.lairez@inria.fr; Aleksandr Storozhenko, École polytechnique, Palaiseau 91120, France, aleksandr.storozhenko@polytechnique.edu.

computational tools. In summary, we begin by constructing an automaton to recognize splittings (Theorem 3). From this, we derive a transducer capable of extracting the atoms of a given word. Using this transducer, we generate an automata that recognizes all possible atoms after a specified number of derivations. Notably, we find that the automata after 24 and 25 derivations are identical (Theorem 5), indicating a convergence in the atom structure, thereby proving the cosmological theorem. This approach involves working with automata and transducers that can have several thousand states.

2 BASICS OF AUTOMATA THEORY

In this section, we present the key elements of automata theory that will be employed to demonstrate the cosmological theorem. For an in-depth review of automata theory, we refer to [5, 9, 8, 7], for example.

2.1 Transducers

Let us define an *alphabet* to be a finite set, calling its elements *symbols*. A *word* over an alphabet A is a finite sequence of symbols of A. A *subword* is a contiguous subsequence of a word. The set of words over A is denoted A^*, and a *language* over an alphabet A is a subset of A^*. The empty word is denoted ε, so we make sure that ε never denotes a symbol of the alphabet. We further define $A^? = A \cup \{\varepsilon\}$, the alphabet augmented with ε, denoting the absence of a symbol.

A *transducer* is a finite directed graph whose edges may be labelled by an input and/or an output symbol, and whose states may be labelled with an *initial* and/or *final* tag. More formally, a transducer is a tuple $\mathcal{T} = (Q, A_{\text{in}}, A_{\text{out}}, E, Q_{\text{initial}}, Q_{\text{final}})$ where:

- Q is the finite set of *states*;
- A_{in} and A_{out} are the *input* and *output* alphabets, respectively;
- $E \subseteq Q \times A_{\text{in}}^? \times A_{\text{out}}^? \times Q$ is the set of transitions, made of a source state, an input symbol (or ε), an output symbol (or ε), and a target state;
- $Q_{\text{initial}} \subseteq Q$ and $Q_{\text{final}} \subseteq Q$ are the sets of *initial* and *final states*, respectively.

A transducer $\mathcal{T}$ defines a *transduction relation*; that is, a binary relation between A^* and B^*, denoted $\to_{\mathcal{T}}$. We say that $u \to_{\mathcal{T}} v$ if there is a path in the graph of the transducer $\mathcal{T}$ from an initial state to a final state, such that the concatenation of the input (respectively output) symbols at the edges along the path is equal to u (respectively v). We say that u is the *input word* and v is the *output word*. If we want to interpret the transducer as a machine reading some input—transitioning from one state to another after each symbol, and producing output on transitions—it is a *nondeterministic* machine: for a given input word, there may be zero, one, finitely or infinitely many execution paths and output words. This nondeterminism will be used extensively. The *input language* of $\mathcal{T}$, denoted $L_{\text{in}}(\mathcal{T})$, is the set of all $u \in A_{\text{in}}^*$ such that $u \to_{\mathcal{T}} v$ for some $v \in A_{\text{out}}^*$. The *output language* of $\mathcal{T}$, denoted $L_{\text{out}}(\mathcal{T})$, is the set of all $v \in A_{\text{out}}^*$ such that $u \to_{\mathcal{T}} v$ for some $u \in A_{\text{in}}^*$. We say that $\mathcal{T}$ *accepts* (respectively *rejects*) a word $x \in A_{\text{in}}^*$ if it belongs (respectively does not belong) to $L_{\text{in}}(\mathcal{T})$.

Different transducers $\mathcal{S}$ and $\mathcal{T}$ may induce the same transduction relation. In this case, we say that they are *equivalent* and write $\mathcal{S} \equiv \mathcal{T}$. Note that the equivalence of transducers is undecidable, meaning it cannot be verified by any finite-time algorithm [7, §**3**.5].

2.2 Useful examples of transducers

Figures 1, 2, 3 and 4 show examples of transducers that will be useful in the proof of the cosmological theorem, and that we describe below. Let A be an alphabet not containing the symbol $\diamond$, and let $B = A \cup \{\diamond\}$.

The "multimark" transducer, denoted *Multi* (Figure 1), works with the input alphabet A and output alphabet B. It inserts arbitrarily many $\diamond$ symbols nondeterministically in the input word. The relation induced by this transducer is characterized as follows: $u \to_{Multi} v$ if and only if u can be obtained from v by deleting the $\diamond$ symbols. For example, $312 \to_{Multi} 3\diamond 1\diamond\diamond 2\diamond\diamond$.

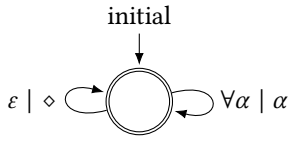

Fig. 1. The "multimark", transducer, denoted *Multi*. The edges are labelled with the convention "input symbol | output symbol". The notation $\forall\alpha$ for the input symbol means the corresponding edge should be duplicated for each symbol in the input alphabet. When the output symbol is α, it means "copy the input symbol". The "initial" arrow marks initial state(s). The double stroke marks final state(s).

The "single mark" transducer, denoted *Mark* (Figure 2), works with the input alphabet A and the output alphabet B. It nondeterministically inserts a $\diamond$ symbol somewhere in the input word, not before the first symbol, and not after the last one. It also accepts the empty word. The relation induced by this transducer is characterized as follows: $\varepsilon \to_{Mark} \varepsilon$ and $uv \to_{Mark} u\diamond v$ for any two nonempty words u and v not containing $\diamond$.

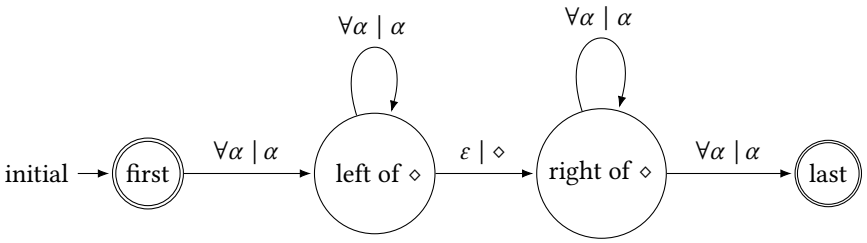

Fig. 2. The "single mark" transducer, denoted *Mark*.

The "scissors" transducer, denoted *Scissors* (Figure 3), works with the input alphabet B and the output alphabet A. It extracts from the input word a substring delimited by $\diamond$ symbols. The relation induced by this transducer is characterized by $u\diamond v\diamond w \to_{Scissors} v$ for all words $u, w \in B^*$ and $v \in A^*$.

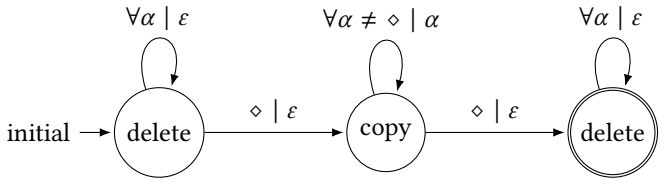

Fig. 3. The "scissors" transducer, denoted *Scissors*.

Lastly, given $a \in A$, the "bounded a-counter" transducer, denoted Cnt_a, works with the input alphabet A and the output alphabet $A \cup \{1, 2, 3\}$. It counts occurrences of a, up to 3. More precisely, the relation induced by this transducer is finite and contains only the following ordered pairs: $a \to_{Cnt_a} 1a$, $aa \to_{Cnt_a} 2a$, and $aaa \to_{Cnt_a} 3a$.

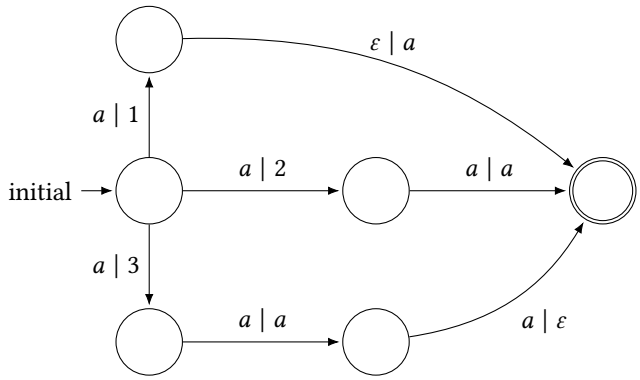

Fig. 4. The "bounded a-counter" transducer, denoted Cnt_a.

2.3 Composition of transducers

Let A, B, and C be alphabets. Let $\mathcal{U}$ be a transducer from the alphabet A to B, and let $\mathcal{V}$ be a transducer from B to C. We define, up to equivalence $\equiv$, a composed transducer $\mathcal{W}$, such that for all $x \in A^*$ and $y \in C^*$,

$$x \to_{\mathcal{W}} y \;\Leftrightarrow\; \exists z \in B^*, x \to_{\mathcal{U}} z \text{ and } z \to_{\mathcal{V}} y,$$

see [7, Theorem **3**.2.2]. We denote $\mathcal{W}$ as $\mathcal{V} \circ \mathcal{U}$. If $\mathcal{U}$ and $\mathcal{V}$ induce partial functions (meaning that there is at most one output word for a given input word), the composition $\mathcal{V} \circ \mathcal{U}$ also induces a partial function which is the composition of the previous ones. The powering notation $\mathcal{U}^n$ denotes the n-fold composition $\mathcal{U} \circ \cdots \circ \mathcal{U}$. A detailed description of the construction is, subsequently, given in the implementation section [4].

2.4 Generators, recognizers, and filters

Let $\mathcal{T}$ be a transducer with an input alphabet A and an output alphabet B. If the input alphabet A is empty, we call $\mathcal{T}$ a *generator*. It is easy to see that the input language of $\mathcal{T}$ corresponds to the singleton $\{\varepsilon\}$. Consequently, on the unique input ε, the transducer generates the complete output language. An example of a *generator* is the "source" transducer *Src*, producing B^* (Figure 5). Similarly, an empty output alphabet B implies the only possible output word is ε. In this case, we call $\mathcal{T}$ a *recognizer*: on every input $u \in A^*$, $\mathcal{T}$ either rejects u, or accepts it with output word ε. An example is given by the "sink" transducer *Sink*, recognizing A^*. Finally, if $A = B$, and if on each transition of $\mathcal{T}$ the input matches the output symbols, we say that $\mathcal{T}$ is a *filter*: on input u, $\mathcal{T}$ either rejects u, or accepts it with output word u.

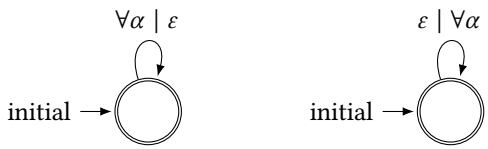

Fig. 5. The "sink" *Sink* and "source" *Src* transducers.

We can convert a recognizer or generator into a filter, and vice versa. For example, if $\mathcal{T}$ is a recognizer, its transitions have the form $(q_1, a, \varepsilon, q_2)$, which replaced by (q_1, a, a, q_2) turn $\mathcal{T}$ into a

filter with the same input language. Furthermore, the composition $Sink \circ \mathcal{T}$ "deletes" the output of $\mathcal{T}$, turning it into a recognizer for $L_{\text{in}}(\mathcal{T})$; similarly, the composition $\mathcal{T} \circ Src$ has the effect of feeding $\mathcal{T}$ all possible inputs, yielding a generator for $L_{\text{out}}(\mathcal{T})$.

The transduction relation of generators (respectively recognizers and filters) is entirely characterized by the output (respectively input) language. These kinds of transducers are essentially the same concept: the concept of *automata*. (In the terminology of [5], our automata are "nondeterministic finite automata with ε-transitions", or ε-NFA). Contrary to general transducers, the equivalence of automata is decidable (see below). This is a key property of our proof.

2.5 Minimal deterministic recognizers

A recognizer $\mathcal{T}$ is *deterministic* if:

- it has a single initial state;
- it has no ε-input transitions;
- for a given state s and a given symbol a, there is at most one transition from s with the input symbol a.

This corresponds to the usual definition of the *deterministic finite automaton* (DFA). Every recognizer is equivalent (in the sense of $\equiv$) to a deterministic recognizer through the power set construction [5, §2.3.5]. Moreover, among all deterministic recognizers of a given language $L \subseteq A^*$, there exists a unique one with the minimal number of states, up to state relabeling [5, §4.4.4]. Given a recognizer for L, there exist various algorithms to compute the associated minimal recognizer. We implemented Brzozowski's algorithm [1] [7, Chapter 10] because of its simplicity. Due to the uniqueness of minimal automata, we can decide the equivalence of two recognizers, $\mathcal{T}$ and $\mathcal{T}'$, by checking the equivalence of their corresponding minimal recognizers.

3 THE COSMOLOGICAL THEOREM

The first step in proving the cosmological theorem is the formulation of the audioactive derivation, in terms of a transducer. Through the subsequent study of *splittings*, the proof becomes a low-hanging fruit.

3.1 The audioactive transducer

Let $\mathbb{N}_>$ denote the set of positive integers and let $\mathbb{N}_>^*$ be the set of all finite sequences over $\mathbb{N}_>$. Audioactive derivation is then a map $C : \mathbb{N}_>^* \to \mathbb{N}_>^*$. A sequence obtained after n applications of C, denoted as $x \in C^n(\mathbb{N}_>^*)$, is called a day-$n$ sequence.

The map C cannot be induced by a transducer, as the associated input and output alphabets, $\mathbb{N}_>$, are not finite. Besides this trivial reason, a transducer has a finite number of states, and thus cannot count to arbitrarily large values. We remark, however, that we can make use of the "one-day theorem" [2, p. 10].

Theorem 1 (One-Day Theorem)**.** *No day-one sequence $x \in C(\mathbb{N}_>^*)$ contains four consecutive equal symbols, that is no "aaaa" subword.*

Proof. By definition of the map C, all pairwise consecutive odd positions in x, indexed from 0, must differ. A subword of length four, however, would contain two consecutive equal odd positions. □

After the first audioactive derivation, all subsequent derivations will only need to count up to three. We still have the problem of an infinite alphabet, but it is only apparent. Indeed, we have seen that a day-one word x contains no *aaaa* subword, so its derivation contains no 44, 55, or *aa* subwords with $a \geq 4$, as one of the two symbols comes from counting consecutive occurrences of

the same symbol in x. Therefore, symbols $d \geq 4$ never mutually interact past the first derivation and are simply carried over. As such, we simply denote them d.

We, hence, consider the alphabet $A = \{1, 2, 3, d\}$. Let $W_{\text{day-one}} \subset A^*$ be the set of words not containing any subword of the form $aaaa$. Audioactive derivation, then, induces a map $C : W_{\text{day-one}} \to W_{\text{day-one}}$, entirely described by a transducer (Figure 6) that uses A as both the input and output alphabet. By virtue of the one-day theorem, it is enough to study the iterations of C on $W_{\text{day-one}}$ to establish the cosmological theorem.

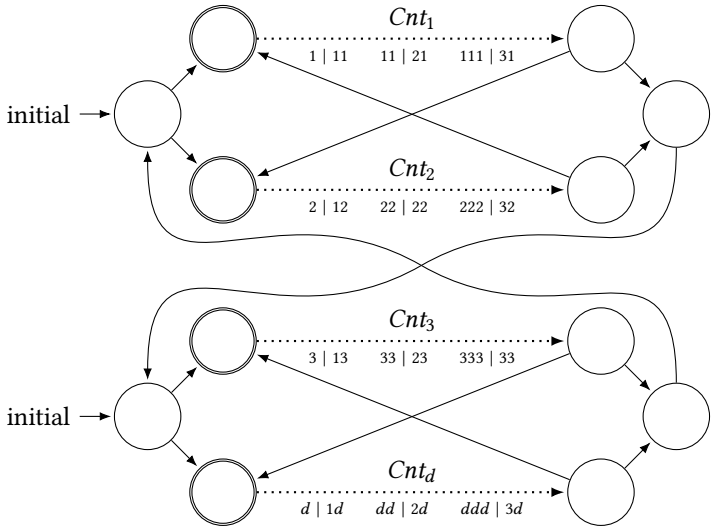

Fig. 6. The "audioactive" transducer, denoted *Audio*, with 28 states. The dotted edges, labelled by a "bounded counter", should be substituted accordingly, identifying the source state with the initial state of the transducer, and the target state with its final state. All other edges have the ε input and output symbols, omitted for legibility. The choice of layout—with a group for the symbols 1 and 2, and another for 3 and d—is only cosmetic, reducing edge crossings.

3.2 Splittings

Let $C^n(x)$ denote the nth audioactive derivation of a word $x \in W_{\text{day-one}}$. A word $x \in W_{\text{day-one}}$ *splits* into $u_1 \cdots u_r$ if $C^n(x) = C^n(u_1) \cdots C^n(u_r)$ for all $n \geq 0$, meaning that the u_i do not interact. This happens exactly when the last digit of $C^n(u_1 \cdots u_i)$ is different from the first digit of $C^n(u_{i+1} \cdots u_r)$ for all $n \geq 0$ and all $1 \leq i < r$. We say that each u_i is a *splitting factor* of x. A nonempty word which admits a single nontrivial splitting factor is an *atom*. A splitting factor which is an atom is called an *atomic factor*. It is easy to check that every nonempty word admits a unique splitting into atoms. For example:

- 32212 splits into $3 \cdot 2212$,[2]
- 32213 splits into $3 \cdot 2213$,
- $3221d$ splits into $3 \cdot 221d$,
- 32211 is an atom,[3]

[2]This is not trivial! This follows from the observation that all the derivations of 3 end with 3, while all the derivations of 2212 start with 2, see Conway's Starting Theorem [2]. We can also check that the word "$3 \diamond 2212$" is accepted by the recognizer *Splitting* introduced below.

[3]The only possible splittings would be $3 \cdot 2211$, which does not work after two derivations, and $322 \cdot 11$, which does not work after one derivation.

- for all $n \geq 1$, the n-time concatenation of 332 is an atom, showing that there are infinitely many atoms.[4]

(The first three assertions can be checked with the *Splitting* automaton, while the last two can be checked with the *Atom* automaton, both introduced below.)

Splitting is subtle. At first glance, there may seem to be infinitely many conditions to check. Yet, we will see that splittings can be recognized by an automaton. To this end, we consider the augmented alphabet $B = \{1, 2, 3, d, \diamond\}$. A word $u_1 \diamond \cdots \diamond u_r$ over B is a *splitting* if $u_1 \cdots u_r \in W_{\text{day-one}}$ and $C^n(u_1 \cdots u_r) = C^n(u_1) \cdots C^n(u_r)$ for all $n \geq 0$. The set of splittings forms a language over B, and we now construct its associated recognizer, *Splitting*.

The "augmented audioactive" transducer, denoted *Audio*$_+$ (Figure 7), extends *Audio* by using B as both the input and output alphabet. Reading $\diamond$ in an accept state, the *Audio*$_+$ transducer outputs the same symbol and remains in the same state. Conversely, when $\diamond$ is read in a non-accepting state, the entire input word is rejected. For example, $22 \diamond 22$ is rejected, but $22 \diamond 33 \rightarrow_{Audio_+} 22 \diamond 23$. The input language of *Audio*$_+$ is the set of words with no *aaaa* subword (with $a \in A$), and no $a \diamond^+ a$ subword (with $a \in A$, where $\diamond^+$ means one or more $\diamond$).

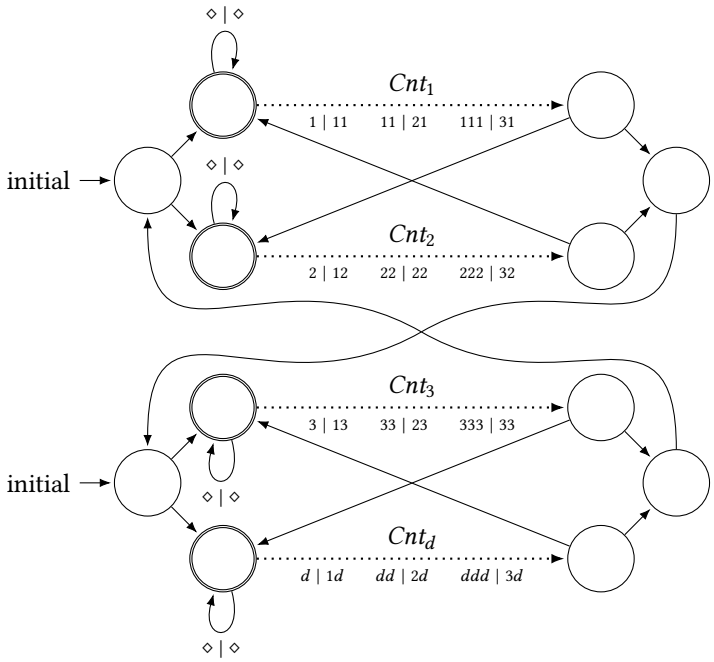

Fig. 7. The augmented audioactive transducer *Audio*$_+$.

Lemma 2. *A word $x \in B^*$ is a splitting if and only if $Audio_+^n$ accepts x for all $n \geq 0$.*

Proof. The condition that $x \in L_{\text{in}}(Audio_+^n)$ for all $n \geq 0$ means that the $\diamond$ symbols (or groups of consecutive $\diamond$ symbols) will never lie between two copies of a symbol of A after any number of audioactive derivations. This is exactly the definition of a splitting. □

Theorem 3 (Splitting Theorem)**.** *The set of splittings is the input language of $Audio_+^9$.*

[4]This follows from the cycle formed by the states w, n, and d in the atom recognizer that we will construct below (Table 3).

Proof. Let L_n denote the input language of $Audio_+^n$, which is the language recognized by $Sink \circ Audio_+^n$. By Lemma 2, the set of splittings is the intersection of all L_n with $n \geq 1$. Since $Audio_+^{n+1} \equiv Audio_+ \circ Audio_+^n$, it follows that $L_{n+1} \subseteq L_n$ for all $n \geq 0$.

We compute $Sink \circ Audio_+^9$ and $Sink \circ Audio_+^{10}$ and verify computationally that they are equivalent. Thus, $Sink \circ Audio_+^n \equiv Sink \circ Audio_+^9$ for all $n \geq 9$, leading to the conclusion that $\cap_{n\geq 1} L_n = L_9$. □

In terms of computation, we compute $Sink \circ Audio_+^n$ using the recurrence relation

$$Sink \circ Audio_+^{n+1} = \left(Sink \circ Audio_+^n\right) \circ Audio_+,$$

minimizing the automata at each step. Given that $Audio_+$ has 28 states, a naive computation of $Sink \circ Audio_+^{10}$ could lead to an automaton with 28^{10} states, likely exhausting the memory of a laptop. However, through iterative minimization, the number of states in $Sink \circ Audio_+^n$ never exceeds 40 (see Table 1), and the computation time is below 10 ms on a standard laptop.

Table 1. Number of states of of $Sink \circ Audio_+^n$ after determinization and minimization.

n	1	2	3	4	5	6	7	8	≥ 9
# states	13	25	37	40	37	29	28	27	21

Following Theorem 3, we define $Splitting \equiv Sink \circ Audio_+^9$, a recognizer for the language of all splittings. After determinization and minimization, *Splitting* consists of 21 states (see Table 2).

Table 2. The splitting recognizer *Splitting*, after determinization and minimization. The set of states is $Q = \{\mathtt{S}, \mathtt{a}, \ldots, \mathtt{t}\}$, the input alphabet is $\{1, 2, 3, d, \diamond\}$.

state	S	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t
initial	•																				
final	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•		•			•	•
input 1	a	b	c		a	a	a	a	a	a	a	a					p	q	c	b	a
input 2	d	d	d	d	e	f		d	d	d	d	d	d	n	o			f		e	d
input 3	g	g	g	g	g	g	g	h	i		g	g	g				r	i			g
input d	j	j	j	j	j	j	j	j	j	j	k	l					s	l		k	t	l
input $\diamond$	S	m	m	m	o	o	o	m	m	m	l	l	l	m			o			m	l	

Conway also provided an explicit description of splittings, which offers an intriguing comparison to our own. Below, we reproduce Conway's splitting theorem [2, p. 11] *verbatim*, intentionally omitting the details of the intricate notations. As Conway himself noted, "this heap of conventions makes it hard to check the proofs, since they cover many more cases than one naively expects." Nevertheless, we offer some insight into how this statement relates to our 21-state automaton, *Splitting*.

The exponent 9 in Theorem 3 is optimal, as shown by the word $3\diamond 133$ which has the following sequence of derivations:

$$3\diamond 133$$
$$13\diamond 1123$$
$$1113\diamond 211213$$
$$3113\diamond 1221121113$$
$$132113\diamond 112221123113$$
$$1113122113\diamond 21322112132113$$
$$311311222113\diamond 1211132221121113122113$$
$$13211321322113\diamond 1112311332211231131122211 3$$
$$1113122113121113222113\diamond 3112132123222112132113213221 13.$$

So this word is accepted by $Audio_+^8$ but not $Audio_+^9$.

The Splitting Theorem. A 2-day-old string LR splits as L.R just if one of L and R is empty or L and R are of the types shown in one of:

L	R
n]	[m (n⩾4, m⩽3)
2]	$[1^1X^1$ or $[1^3$ or $[3^1X^{\neq 3}$ or $[n^1$ (n⩾4)
≠2]	$[2^21^1X^1$ or $[2^21^3$ or $[2^23^1X^{\neq 3}$ or $[2^2n^{(0 \text{ or } 1)}$ (n⩾4)

We briefly outline the structure of the splitting recognizer (Table 2). The states S, a, …, l form a recognizer for the language $W_{\text{day-one}}$. This component of the automaton counts consecutive identical symbols, up to a maximum of three, and rejects all words, containing four or more consecutive equal symbols. For instance, the state c, reached after reading three consecutive 1s, does not accept an additional 1, as it has no transition for the input 1. When *Splitting* encounters the symbol $\diamond$, it transitions to the second part of the automaton, comprised of states l, …, t. This part has three entry points:

- The state l is reached from the states j, k, or l, where the last input received is *d*. This state corresponds to the first line of Conway's statement and only accepts a digit 1, 2, or 3.
- The state o is reached from the states d, e, or f, where the last input received is 2. This state corresponds to the second line of Conway's statement.
- The state m is reached from the states where the last input received is 1 or 3. It corresponds to the third line of Conway's statement. This state only accepts the input 22 and transitions to the state o, reflecting the similarities in the structure of the second and third lines of Conway's statement.

For example, consider the input word $3\diamond 22123$. This matches what Conway denotes as "$\neq 2]\ [2^21^1X^1$." On this input, *Splitting* will transition through the states S, g, m, n, o, p, f, or g. The final state is accepting, indicating that $3\diamond 22123$ constitutes a valid splitting. Next, consider the input word $3\diamond 22122$. This sequence does not match "$\neq 2]\ [2^21^1X^1$" because the final 2 cannot be ignored, as it is repeated. (The conventions are truly subtle.) On this input, *Splitting* will follow the same sequence of states as before, except at state f, which will reject the final input symbol 2.

Therefore, $3\diamond 22122$ is not a valid splitting. We can explicitly verify the derivation sequence:

$$3\diamond 22122 \longrightarrow 13\diamond 221122 \longrightarrow 1113\diamond 222122 \longrightarrow 3113\diamond 321122,$$

where we observe the digits before and after the $\diamond$ are identical, thus violating the splitting condition.

From the *Splitting* recognizer, we can now construct a recognizer for the set of irreducible words. Consider the recognizer $\mathit{Splitting} \circ \mathit{Mark}$ (where *Mark* is the "single mark" transducer in Figure 2), it accepts exactly the empty word, and words uv, such that $u\diamond v$ is a splitting, with u and v nonempty words over A. Put differently, this recognizer rejects the irreducible words in $W_{\text{day-one}}$, accepting all others. Let $\overline{\mathit{Splitting} \circ \mathit{Mark}}$ denote the complement recognizer, easily computable after determinization [5, §4.2]. It accepts exactly the words rejected by $\mathit{Splitting} \circ \mathit{Mark}$, recognizing the set of atoms in $W_{\text{day-one}}$ union the set of words not in $W_{\text{day-one}}$. Finally, we want to also reject the words not in $W_{\text{day-one}}$, which are exactly the words rejected by $\mathit{Sink} \circ \mathit{Audio}$. We, thus, define the irreducible word recognizer

$$\mathit{Atom} = \overline{\mathit{Splitting} \circ \mathit{Mark}} \circ (\mathit{Sink} \circ \mathit{Audio})_{\mathit{filter}},$$

where the subscript "filter" indicates that we turn a recognizer into a filter. This 26-state automaton is shown in Table 3. By construction, we obtain the following statement.

Lemma 4. *The automaton Atom recognizes exactly the set of all atoms in $W_{\text{day-one}}$.*

Table 3. The atom recognizer *Atom*, after determinization and minimization.

state	S	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y
initial	•																									
final		•	•	•	•	•			•						•	•	•	•	•						•	•	•
input 1		a	b	c		e	c		e	e	j	k	c			a	a								a	e	e
input 2		x	d	d	d	g	f	g	h		l			m	h	d	d								d	y	h
input 3		w	w	w	w	i	u		i	i	n					o						n			n	i	i
input *d*		p	p	p	p	t	v		t	t	s					p	p	q	r		t	r		q	p	t	t

3.3 Cosmological Theorem

The introduced notions enable us to formulate a precise statement about audioactive decay. Let E denote the set of all common and transuranic elements. It is a set of 94 atoms of which elements are given in the appendix.

Theorem 5 (Cosmological Theorem)**.** *For every word x over A without "aaaa" subwords, and all $n \geq 24$, the nth audioactive derivation of x splits into atoms belonging to E.*

Let E_n denote the set of all atomic factors of all the words in $L_{\text{out}}(\mathit{Audio}^n)$. Our goal is to show that E_n stabilizes for $n \geq 24$. We further aim to explicitly describe the ultimate value of E_n. The proof rests upon the recognizers *Splitting* and *Atom*.

Let us construct a transducer *AtomicF*, for "atomic factor", over the alphabet A for both input and output, such that $u \longrightarrow_{\mathit{AtomicF}} v$ if and only if v is an atomic factor of u. This transducer is simply

$$\mathit{AtomicF} = \mathit{Atom}_{\mathit{filter}} \circ \mathit{Scissors} \circ \mathit{Splitting}_{\mathit{filter}} \circ \mathit{Multi},$$

where the subscript "filter" indicates the conversion of a recognizer into a filter. More explicitly, the transducer inputs a word over A, and then:

- the multimark transducer *Multi* nondeterministically inserts $\diamond$ symbols;

- the splitting recognizer *Splitting* exclusively retains the splittings;
- the transducer *Scissors* nondeterministically extracts a splitting factor of a given splitting;
- the atom recognizer *Atom* only retains the atomic factors.

Proof of Theorem 5. We compute—see Figure 8 and the subsequent discussion—that

$$AtomicF \circ Audio^{25} \circ Src \equiv AtomicF \circ Audio^{24} \circ Src, \tag{1}$$

so $E_{25} = E_{24}$. Since the atomic factors of the derivation of a word x are exactly the atomic factors of the derivations of the atomic factors of x, it follows that the elements of E_{n+1} are the atomic factors of the derivation of the elements of E_n. Therefore, $E_{25} = E_{24}$ implies $E_n = E_{24}$ for all $n \geq 24$.

It remains to enumerate all the elements of E_{24}, which boils down to the enumeration of all the paths from any of the initial states to any of the final states in $AtomicF \circ Audio^{24} \circ Src$. □

It is possible to check the equivalence in (1) directly, but it is difficult because $Audio^{25} \circ Src$ is a large automaton, with 194,625 states, after minimization. Instead, we observe that $AtomicF \circ Audio \equiv AtomicF \circ Audio \circ AtomicF$, because, as noted in the proof of Theorem 5, the irreducible splitting factors of the derivation of a word x are exactly the irreducible splitting factors of the derivations of the irreducible splitting factors of x. It follows that for any $n \geq 1$,

$$AtomicF \circ Audio^{n} \circ Src = AtomicF \circ Audio \circ \left(AtomicF \circ Audio^{n-1} \circ Src\right).$$

This gives a much more efficient recursive way of computing $AtomicF \circ Audio^{n} \circ Src$. Naturally, we minimize the automata after each composition. The maximum number of states for $AtomicF \circ Audio^{n} \circ Src$ is 592, when $n = 6$. The total computation time is 150 ms on a laptop.

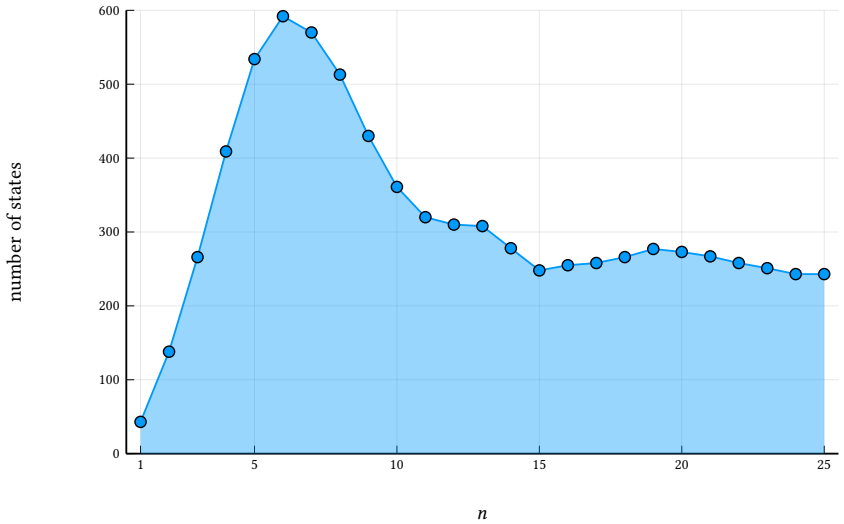

Fig. 8. Number of states of $AtomicF \circ Audio^{n} \circ Src$, after determinization and minimization.

4 IMPLEMENTATION

For readers interested in consulting the code or reproducing the results, we present a brief comment on our C++ implementation. The code is available at

https://github.com/AleksandrStorozhenko/ConwayTransducer.

4.1 Data structure

It is convenient to index the symbols in the input and output alphabets, so that we can assume that they are $\{0, \dots, n-1\}$ and $\{0, \dots, m-1\}$, respectively. We can represent the transitions of a transducer $\mathcal{T}$ as a 2d array T of arrays of pairs such that $T[s][a]$ contains the array of all transitions with source state s and input symbol a. In C++, this gives the following unsophisticated data structure for manipulating transducers.

```
#include <set>
#include <vector>
using namespace std;

struct Transducer {
  using symbol = int;
  using state = int;

  int inputSymbols, outputSymbols;
  set<state> startNodes, finalNodes;
  vector<vector<vector<pair<symbol, state>>>> table;
};
```

The chosen representation does not, however, lend itself to an effective manipulation of generators (with an empty input language), as we cannot efficiently query transitions by output symbol. Instead, we transpose generators into recognizers, swapping input and output symbols for each transition.

4.2 Composition

Given two finite state transducers $\mathcal{U}$ and $\mathcal{V}$, such that the output alphabet of $\mathcal{U}$ matches the input alphabet of $\mathcal{V}$, we want to compute a transducer $\mathcal{W}$ realizing the composion $\mathcal{V} \circ \mathcal{U}$.

We choose the set of states $Q_{\mathcal{W}}$ to be $Q_{\mathcal{U}} \times Q_{\mathcal{V}}$. The initial states of $\mathcal{W}$ are pairs of initial states of $\mathcal{U}$ and $\mathcal{V}$ respectively, and similarly for final states. For each state $(q_u, q_v) \in Q_{\mathcal{U}} \times Q_{\mathcal{V}}$, we declare the following transitions in $\mathcal{W}$:

- $((q_u, q_v), a, \varepsilon, (q'_u, q_v))$ whenever there is a transition $(q_u, a, \varepsilon, q'_u)$ in $\mathcal{U}$;
- $((q_u, q_v), \varepsilon, c, (q_u, q'_v))$ for any transition $(q_v, \varepsilon, c, q'_v)$ in $\mathcal{V}$; and
- $((q_u, q_v), a, c, (q'_u, q'_v))$ whenever there exists transitions (q_u, a, b, q'_u) and (q_v, b, c, q'_v) in $\mathcal{U}$ and $\mathcal{V}$ respectively for some $b \in B$.

In practice, only a fraction of the constructed states is reachable, so it is worthwhile to construct the state set by a traversal from the initial states to avoid the unreachable sets.

4.3 Determinization

A recognizer $\mathcal{T}$ admits an equivalent deterministic recognizer $\mathcal{D}$ with the associated set of states equal to the set of subsets of $Q_{\mathcal{T}}$ (the set of states of $\mathcal{T}$). The initial state of $\mathcal{D}$ corresponds to the set of initial states of $\mathcal{T}$. For $U \in Q_{\mathcal{D}}$ and a symbol a, there exists a transition (U, a, ε, V) in $\mathcal{D}$ for $V \in Q_{\mathcal{D}}$, the set of all states, reachable by a transition in $\mathcal{T}$ from a state in U with an input symbol a. In practice, only a fraction of these states is reachable. We, thus, perform the determinization by a traversal from the initial state. In the worst case, the deterministic recognizer $\mathcal{D}$ admits exponentially many states in the size of $Q_{\mathcal{T}}$.

4.4 Minimization

Within the context of this paper, minimization of deterministic recognizers (and generators *via* transposition) satisfies two principal aims: computational efficiency via reduction of states; and language equivalence verification.

For minimizing recognizers, we resorted to Brzozowski's algorithm [1] [7, Chapter 10]: Given a recognizer $\mathcal{T}$, swap the source and target states of each transitions (reversal), determinize, reverse again, determinize again. The algorithm has a worst-case exponential complexity, due to the potential of exponential growth in the number of states at the first determinization, but it was enough for our purposes.

Algorithms for transducers (minimization, equivalence) are more subtle, with many undecidability results [7, Chapter 3]. Yet, we found it useful to have a heuristic size-reduction procedure by interpreting a transducer over input alphabet A and output alphabet B as a recognizer over the alphabet $A^? \times B^?$, simply considering the input-output pair of each transition as an input symbol, and minimizing it. For example, the size of the transducer $AtomicF \circ Audio$ reduces from 977 states to 82. This makes the computation of $(AtomicF \circ Audio)^k \circ Src$ much faster in the proof of Theorem 5.

4.5 Equivalence of recognizers

To check that two recognizers $\mathcal{T}$ and $\mathcal{T}'$ are equivalent, it is enough to check that the minimization of $\mathcal{T}$ and $\mathcal{T}'$ [5, §4.4] are equal, up to a bijection $Q_{\mathcal{T}} \to Q_{\mathcal{T}'}$. We can construct this bijection, or prove that it does not exist, by a traversal from the initial states of $\mathcal{T}$ and $\mathcal{T}'$.

CONCLUSION

The study of audioactive decay, from Conway's discovery to the four known computational proofs, including this one, is an outstanding illustration of the principles of *experimental mathematics*. It was, of course, experimentation that led to the formulation of the cosmological theorem, before any proof could be provided. But the experimental approach in mathematics extends far beyond mere data accumulation for conjecture formation. Our entire automaton-based proof is experimental in nature. It is not a proof *by computation*, but a proof *by experimentation*. While there is an algorithm that proves that $3 \times 19 = 57$, for instance, we do not have an algorithm that proves the cosmological theorem. Computation serves as a microscope or spectrometer, enabling us to observe and analyze mathematical phenomena without prior knowledge of their nature. The nature of the cosmological theorem is not a consequence of the formulation of the audioactive derivation by a transducer (choose a different transducer, and you might not observe a splitting theorem or a cosmological theorem), but experimentation and computation uncover this structure.

ACKNOWLEDGMENTS

This work has been supported by the Agence nationale de la recherche (ANR), grant agreement ANR-19-CE40-0018 (De Rerum Natura); and by the European Research Council (ERC) under the European Union's Horizon Europe research and innovation programme, grant agreement 101040794 (10000 DIGITS).

REFERENCES

1. Brzozowski, J.: Canonical regular expressions and minimal state graphs for definite events. Mathematical Theory of Automata **12**, 529–561 (1962)
2. Conway, J.H.: The Weird and Wonderful Chemistry of Audioactive Decay. Eureka **46**, 5–16 (1986)
3. Conway, J.H.: The Weird and Wonderful Chemistry of Audioactive Decay. In: Open Problems in Communication and Computation. Ed. by T.M. Cover and B. Gopinath, pp. 173–188. Springer, New York, NY (1987). https://doi.org/10/dpwfr6

4. Ekhad, S.B., Zeilberger, D.: Proof of Conway's Lost Cosmological Theorem. Electron. Res. Announc. Amer. Math. Soc. **3**(11), 78–82 (1997). https://doi.org/10/d4n7zp
5. Hopcroft, J.E., Motwani, R., Ullman, J.D.: Introduction to Automata Theory, Languages, and Computation. Pearson, Addison-Wesley (2007)
6. Litherland, R.A.: Conway's Cosmological Theorem, (2003). https://oeis.org/A005150/a005150_3.pdf.
7. Pin, J.-É. (ed.): Handbook of Automata Theory. EMS (2021)
8. Sakarovitch, J.: Elements of Automata Theory. Cambridge University Press, Cambridge (2009)
9. Shallit, J.: A Second Course in Formal Languages and Automata Theory. Cambridge University Press (2008)
10. Watkins, K.: Abstract Interpretation Using Laziness: Conway's Lost Cosmological Theorem, (2006). https://www.cs.cmu.edu/~kw/pubs/conway.pdf.

APPENDIX: PERIODIC TABLE

name	derivation	element
$_{1}$H	$_{1}$H	22
$_{2}$He	$_{72}$Hf $_{91}$Pa $_{1}$H $_{20}$Ca $_{3}$Li	13112221133211322112211213322112
$_{3}$Li	$_{2}$He	312211322212221121123222112
$_{4}$Be	$_{32}$Ge $_{20}$Ca $_{3}$Li	1113122113121132211332113221122112113322112
$_{5}$B	$_{4}$Be	13211321222113222122211211232222112
$_{6}$C	$_{5}$B	311311221132211221121332112
$_{7}$N	$_{6}$C	11131221222112112322112
$_{8}$O	$_{7}$N	132112211213322112
$_{9}$F	$_{8}$O	31121123222112
$_{10}$Ne	$_{9}$F	111213322112
$_{11}$Na	$_{10}$Ne	123222112
$_{12}$Mg	$_{61}$Pm $_{11}$Na	3113322112
$_{13}$Al	$_{12}$Mg	1113222112
$_{14}$Si	$_{13}$Al	1322112
$_{15}$P	$_{67}$Ho $_{14}$Si	311311222112
$_{16}$S	$_{15}$P	1113122112
$_{17}$Cl	$_{16}$S	132112
$_{18}$Ar	$_{17}$Cl	3112
$_{19}$K	$_{18}$Ar	1112
$_{20}$Ca	$_{19}$K	12
$_{21}$Sc	$_{67}$Ho $_{91}$Pa $_{1}$H $_{20}$Ca $_{27}$Co	3113112221133112
$_{22}$Ti	$_{21}$Sc	11131221131112
$_{23}$V	$_{22}$Ti	13211312
$_{24}$Cr	$_{23}$V	31132
$_{25}$Mn	$_{24}$Cr $_{14}$Si	111311222112
$_{26}$Fe	$_{25}$Mn	13122112
$_{27}$Co	$_{26}$Fe	32112
$_{28}$Ni	$_{30}$Zn $_{27}$Co	11133112
$_{29}$Cu	$_{28}$Ni	131112
$_{30}$Zn	$_{29}$Cu	312
$_{31}$Ga	$_{63}$Eu $_{20}$Ca $_{89}$Ac $_{1}$H $_{20}$Ca $_{30}$Zn	13221133122211332
$_{32}$Ge	$_{67}$Ho $_{31}$Ga	311311222113111221132222
$_{33}$As	$_{32}$Ge $_{11}$Na	1113122113121132211332112
$_{34}$Se	$_{33}$As	13211321222113222112
$_{35}$Br	$_{34}$Se	3113112211322112

$_{36}$Kr	$_{35}$Br	11131221222112
$_{37}$Rb	$_{36}$Kr	1321122112
$_{38}$Sr	$_{37}$Rb	3112112
$_{39}$Y	$_{38}$Sr $_{92}$U	1112133
$_{40}$Zr	$_{39}$Y $_{1}$H $_{20}$Ca $_{43}$Tc	123222113312221131122 11
$_{41}$Nb	$_{68}$Er $_{40}$Zr	1113122113322113111221131221
$_{42}$Mo	$_{41}$Nb	13211322211312113211
$_{43}$Tc	$_{42}$Mo	311322113212221
$_{44}$Ru	$_{63}$Eu $_{20}$Ca $_{43}$Tc	132211331222113112211
$_{45}$Rh	$_{67}$Ho $_{44}$Ru	3113112221131112211312 21
$_{46}$Pd	$_{45}$Rh	111312211312113211
$_{47}$Ag	$_{46}$Pd	132113212221
$_{48}$Cd	$_{47}$Ag	3113112211
$_{49}$In	$_{48}$Cd	11131221
$_{50}$Sn	$_{49}$In	13211
$_{51}$Sb	$_{61}$Pm $_{50}$Sn	3112221
$_{52}$Te	$_{63}$Eu $_{20}$Ca $_{51}$Sb	1322113312211
$_{53}$I	$_{67}$Ho $_{52}$Te	311311222113111221
$_{54}$Xe	$_{53}$I	11131221131211
$_{55}$Cs	$_{54}$Xe	13211321
$_{56}$Ba	$_{55}$Cs	311311
$_{57}$La	$_{56}$Ba	11131
$_{58}$Ce	$_{57}$La $_{1}$H $_{20}$Ca $_{27}$Co	1321133112
$_{59}$Pr	$_{58}$Ce	31131112
$_{60}$Nd	$_{59}$Pr	111312
$_{61}$Pm	$_{60}$Nd	132
$_{62}$Sm	$_{61}$Pm $_{20}$Ca $_{30}$Zn	311332
$_{63}$Eu	$_{62}$Sm	1113222
$_{64}$Gd	$_{63}$Eu $_{20}$Ca $_{27}$Co	13221133112
$_{65}$Tb	$_{67}$Ho $_{64}$Gd	3113112221131112
$_{66}$Dy	$_{65}$Tb	111312211312
$_{67}$Ho	$_{66}$Dy	1321132
$_{68}$Er	$_{67}$Ho $_{61}$Pm	311311222
$_{69}$Tm	$_{68}$Er $_{20}$Ca $_{27}$Co	11131221133112
$_{70}$Yb	$_{69}$Tm	1321131112
$_{71}$Lu	$_{70}$Yb	311312
$_{72}$Hf	$_{71}$Lu	11132
$_{73}$Ta	$_{72}$Hf $_{91}$Pa $_{1}$H $_{20}$Ca $_{74}$W	13112221133211322112211213322113
$_{74}$W	$_{73}$Ta	312211322212221121123222113
$_{75}$Re	$_{32}$Ge $_{20}$Ca $_{74}$W	1113122113121132211332113221122112132221 13
$_{76}$Os	$_{75}$Re	132113212221132221222112112322211 3
$_{77}$Ir	$_{76}$Os	31131122113221122112133221 13
$_{78}$Pt	$_{77}$Ir	1113122122211211232221 13
$_{79}$Au	$_{78}$Pt	132112211213322113
$_{80}$Hg	$_{79}$Au	31121123222113
$_{81}$Tl	$_{80}$Hg	111213322113
$_{82}$Pb	$_{81}$Tl	123222113
$_{83}$Bi	$_{61}$Pm $_{82}$Pb	3113322113

$_{84}$Po	$_{83}$Bi	1113222113
$_{85}$At	$_{84}$Po	1322113
$_{86}$Rn	$_{67}$Ho $_{85}$At	311311222113
$_{87}$Fr	$_{86}$Rn	1113122113
$_{88}$Ra	$_{87}$Fr	132113
$_{89}$Ac	$_{88}$Ra	3113
$_{90}$Th	$_{89}$Ac	1113
$_{91}$Pa	$_{90}$Th	13
$_{92}$U	$_{91}$Pa	3
(transuranic elements)		
$_{93}$Np	$_{72}$Hf $_{91}$Pa $_{1}$H $_{20}$Ca $_{94}$Pu	13112221133211322112211213322114
$_{94}$Pu	$_{93}$Np	31221132221222112112322211*d*